PUCRS ESCOLA POLITÉCNICA

Projeto e Otimização de Algoritmos

Algoritmos Gulosos

MARISTA

Prof. Rafael Scopel

Algoritmos Gulosos (Greedy)

- Greed(y) is good!

- Nesta aula vamos investigar os pontos fortes e fracos da analise greedy no desenho de algoritmos.

- Para isso vamos abordar diferentes problemas computacionais com as seguintes perguntas: Greedy é bom? Greedy funciona?

- É difícil definir precisamente o que queremos dizer com um algoritmo greedy.

- Um algoritmo é greedy se ele constrói uma solução com pequenos passos, fazendo uma escolha local (míope) a cada passo para otimizar um determinado critério ou função objetivo.

- É possível desenhar diversos algoritmos greedy para um mesmo problema, onda cada algoritmo implementa um passo local míope que vai otimizando uma medida diferente de performance no seu caminho para a solução.

- Quando um algoritmo greedy tem sucesso em resolver um problema não-trivial, significa que podemos obter soluções ótimas a partir de decisões locais.

Algoritmos Gulosos (Greedy): Troco em moedas

- Todo o caixa de supermercado precisa gerenciar as suas moedas para troco da forma mais eficiente o possível.
- Considerando as moedas existentes no Brasil {1, 5, 10, 25, 50 ,100}, **defina um método para pagar o cliente utilizando a menor quantidade de moedas.**

PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Troco em moedas

- Qual a solução para um troco de R$ 0.36

- Várias soluções:

Algoritmos Gulosos (Greedy): Troco em moedas

O ALGORITMO (Cashiers-Algorithm):

- Um algoritmo greedy para resolver o valor de troco n em moedas, funciona da seguinte maneira:

1. Separe $g = \lfloor n/100 \rfloor$ moedas de 1 real. Agora temos um novo valor de troco residual $n_g = n - g = n \,\%\, 100$.
2. Separe $h = \lfloor n_g/50 \rfloor$ moedas de 50 centavos. Agora temos um novo valor de troco residual $n_h = n_g - h = n_g \% \, 50$.
3. Separe $\mathrm{q} = \lfloor n_h/25 \rfloor$ moedas de 25 centavos. Agora temos um novo valor de troco residual $n_q = n_h - q = n_h \% \, 25$.
4. Separe $\mathrm{d} = \lfloor n_q/10 \rfloor$ moedas de 10 centavos. Agora temos um novo valor de troco residual $n_d = n_q - q = n_q \% \, 10$.
5. Separe $\mathrm{k} = \lfloor n_d/5 \rfloor$ moedas de 5 centavos. Agora temos um novo valor de troco residual $n_k = n_d - k = n_d \% \, 5$.
6. Finalmente, separe $p = n_k$ moedas de 1 centavo.

Testando com $R\$\ 0.36$, ou seja, $n = 36$ e um conjunto de troco $S = \emptyset$ inicialmente vazio.

1. Separamos g$= \lfloor 36/100 \rfloor = 0$ moedas de 50 centavos. $n_g = 36 - 0 = 36 \,\%\, 100 = 36$.
2. Separamos $h = \lfloor 36/50 \rfloor = 0$ moedas de 50 centavos. $n_h = 36 - 0 = 36 \,\%\, 50 = 36$.
3. Separamos $\mathrm{q} = \lfloor 36/25 \rfloor = 1$ moeda de 25 centavos. Agora temos um novo valor de troco residual $n_q = 36 - 25 = 36 \,\%\, 25 =$11. Adicione o troco no conjunto, $S = \{25\}$.
4. Separamos $\mathrm{d} = \lfloor 11/10 \rfloor = 1$ moeda de 10 centavos. Agora temos um novo valor de troco residual $n_d = 11 - 10 = 11 \,\%\, 10 = 1$. Adicione o troco no conjunto, $S = \{25{,}10\}$.
5. Separamos $\mathrm{k} = \lfloor 1/5 \rfloor = 0$ moedas de 5 centavos. Agora temos um novo valor de troco residual $n_k = 1 - 0 = 1 \,\%\, 5 = 1$.
6. Finalmente, separamos $p = n_k = 1$ moeda de 1 centavo. Adicione o troco no conjunto, $S = \{25{,}10, 1\}$.

PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Troco em moedas

Um algoritmo greedy para resolver o valor de troco n em moedas, funciona da seguinte maneira:

1. Separe $g = \lfloor n/100 \rfloor$ moedas de 1 real. Agora temos um novo valor de troco residual $n_g = n - g = n\,\%\,100$.
2. Separe $h = \lfloor n_g/50 \rfloor$ moedas de 50 centavos. Agora temos um novo valor de troco residual $n_h = n_g - h = n_g\%\,50$.
3. Separe q $= \lfloor n_h/25 \rfloor$ moedas de 25 centavos. Agora temos um novo valor de troco residual $n_q = n_h - q = n_h\%\,25$.
4. Separe d $= \lfloor n_q/10 \rfloor$ moedas de 10 centavos. Agora temos um novo valor de troco residual $n_d = n_q - q = n_q\%\,10$.
5. Separe k $= \lfloor n_d/5 \rfloor$ moedas de 5 centavos. Agora temos um novo valor de troco residual $n_k = n_d - k = n_d\%\,5$.
6. Finalmente, separe $p = n_k$ moedas de 1 centavo.

CASHIERS-ALGORITHM $(x, c_1, c_2, \ldots, c_n)$

SORT n coin denominations so that $0 < c_1 < c_2 < \ldots < c_n$.

$S \leftarrow \varnothing$. ⟵ multiset of coins selected

WHILE $(x > 0)$

  $k \leftarrow$ largest coin denomination c_k such that $c_k \leq x$.

  IF (no such k)

    RETURN *"no solution."*

  ELSE

    $x \leftarrow x - c_k$.

    $S \leftarrow S \cup \{ k \}$.

RETURN S.

Algoritmos Gulosos (Greedy): Troco em moedas

ANALISANDO O ALGORITMO:

- Um algoritmo greedy obtém uma solução ótima para um problema fazendo uma sequência de escolhas.
- A cada ponto de decisão, o algoritmo faz um decisão que parece a melhor no momento.
- Esta estratégia heurística nem sempre produz uma solução ótima, mas quando um algoritmo greedy tem sucesso em resolver um problema não-trivial, significa que podemos obter soluções ótimas a partir de decisões locais.
- Como podemos afirmar que um algoritmo greedy vai resolver um determinado problema?
- Para garantir que um algoritmo greedy resolve um determinado problema precisamos procurar por dois ingredientes chave:
  - A propriedade da escolha greedy
  - A subestrutura ótima do problema.
- Se conseguirmos demonstrar esses dois ingredientes estamos no caminho certo para a validade de uma solução greedy para o nosso problema.

Algoritmos Gulosos (Greedy): Troco em moedas

A propriedade da escolha greedy

- Podemos obter uma solução ótima global realizando escolhas míopes (greedy) locais.
- Em outras palavras, quando consideramos que escolha fazer, sempre escolhemos que parece a melhor no problema atual, sem considerar o resultado de subproblemas.
- Ou seja, resolvemos a problema atual e depois resolvemos os problemas remanescentes (primeiro tínhamos que dar troco para R$0.36 e tomamos a melhor decisão possível, depois tínhamos R$0.11 e tomamos a melhor decisão possível, ...).
- As escolhas feitas pelo algoritmo greedy dependem das decisões greedy anteriores, mas não das decisões futuras ou das soluções de outros subproblemas.
- Podemos dizer que o algoritmo greedy funciona top-down, fazendo uma decisão greedy após a outra reduzindo a instância do problema a cada decisão (por exemplo, um menor valor de troco).
- Precisamos provar que a decisão greedy a cada passo sempre resulta num solução global ótima.
- Podemos fazer isso por indução ou substituição como veremos mais a frente.

Algoritmos Gulosos (Greedy): Troco em moedas

A subestrutura ótima do problema.

- Um problema possui uma subestrutura ótima se uma solução ótima para o problema contém dentro dela soluções ótimas para os subproblemas.
- Por exemplo, suponha que temos uma solução ótima S^* para o problema de fazer troco para n centavos, e sabemos que essa solução ótima utiliza uma moeda de valor c centavos.
- Assuma que essa solução ótima S^* use k moedas.
- Afirmamos que essa solução ótima para o problema de n centavos deve conter dentro dela uma solução ótima para o problema de troco para $n - c$ centavos.
- Para provar esta afirmação utilizamos o argumento de substituição (cut-and-paste).
- Claramente, existem $k - 1$ moedas na solução para o problema de $n - c$ centavos que foram utilizadas para a nossa solução do problema de n centavos.
- Se tivéssemos uma solução para o problema de $n - c$ centavos que utilizasse menos de $k - 1$ moedas, então poderíamos utilizar essa solução para produzir uma solução para o problema de n centavos que utiliza menos de k moedas, o que contradiz a nossa solução ótima **(LEMBRE-SE <u>ASSUMIMOS</u> QUE A SOLUÇÃO ÓTIMA USA $\boldsymbol{k}$ MOEDAS).**

Algoritmos Gulosos (Greedy): Troco em moedas

- Para provar que o algoritmo de Cashiers resulta numa solução ótima, primeiramente precisamos demostrar que a propriedade de escolha greedy é respeitada.
- Ou seja, que alguma solução ótima para fazer troco para n centavos inclui uma moeda de valor c, onde c é o maior valor de moeda de forma que $c \leq n$ (decisão greedy!!!).
- Considere alguma solução ótima S^*.
- Se essa solução ótima inclui uma moeda de valor c, então a prova está concluída.
- Caso contrário, esta solução não inclui a moeda de valor c e temos que considerar 6 casos:

1. Se $1 \leq n < 5$, então $c = 1$. Uma solução pode consistir de apenas moedas de 1 centavo e, portanto, deve conter a estratégia greedy.

2. Se $5 \leq n < 10$, então $c = 5$. Por suposição, essa solução não contém uma moeda de 5 centavos, e portanto possui apenas moedas de 1 centavo. Substituímos 5 moedas de 1 centavos por uma de cinco centavos para entregar uma solução com 4 moedas a menos.

3. Se $10 \leq n < 25$, então $c = 10$. Por suposição, essa solução não contém uma moeda de 10 centavos, e portanto possui apenas moedas de 1 e/ou 5 centavos. Algum subconjunto de moedas de 1 e 5 centavos soma 10 centavos, então podemos substituir esse subconjunto por uma moeda de 10 centavos e entregar uma solução com 1 a 9 moedas a menos.

4. Se $25 \leq n < 50$, então $c = 25$. Por suposição, essa solução não contém uma moeda de 25 centavos, e portanto possui apenas moedas de 1, 5 e/ou 10 centavos. Algum subconjunto de moedas de 1, 5 e 10 centavos soma 25 centavos, então podemos substituir esse subconjunto por uma moeda de 25 centavos e entregar uma solução com 1 a 24 moedas a menos.

Algoritmos Gulosos (Greedy): Troco em moedas

5. Se $50 \leq n < 100$, então $c = 50$. Por suposição, essa solução não contém uma moeda de 50 centavos, e portanto possui apenas moedas de 1, 5, 10 e/ou 25 centavos. Algum subconjunto de moedas de 1, 5, 10 e 25 centavos soma 50 centavos, então podemos substituir esse subconjunto por uma moeda de 50 centavos...
6. Se $100 \leq n$, então $c = 100$. Por suposição, essa solução não contém uma moeda de 100 centavos, e portanto possui apenas moedas de 1, 5, 10, 25 e/ou 50 centavos. Algum subconjunto de moedas de 1, 5, 10, 25 e 50 centavos soma 100 centavos, então podemos substituir esse subconjunto por uma moeda de 100 centavos...

- Logo, demonstramos que sempre existe uma solução ótima que inclui a escolha greedy, e que podemos combinar a escolha greedy com uma solução ótima para os próximos subproblemas e produzir uma solução ótima global para o nosso problema original.
- Portanto, o algoritmo greedy Cashiers produz uma solução ótima.

- O algoritmo greedy sempre produz uma solução ótima para todas as denominações de moedas?
- Imagine se o Real só possui-se as seguintes moedas 1 centavo, 10 centavos e 25 centavos.
- Use o algoritmo do Cashier para dar troco para 30 centavos?

Algoritmos Gulosos (Greedy): Troco em moedas

- O algoritmo greedy sempre produz uma solução ótima para todas as denominações de moedas?
- Imagine se o Real só possui-se as seguintes moedas 1 centavo, 10 centavos e 25 centavos.
- Use o algoritmo do Cashier para dar troco para 30 centavos?

Algoritmos Gulosos (Greedy): Troco em moedas

- O algoritmo greedy sempre produz uma solução ótima para todas as denominações de moedas?
- Imagine se o Real só possui-se as seguintes moedas 1 centavo, 10 centavos e 25 centavos.
- Use o algoritmo do Cashier para dar troco para 30 centavos?
  - 1 moeda de 25 centavos
  - 5 moedas de 1 centavo
- Total de 6 moedas. Essa solução é ótima?
- Claramente não, pois 3 moedas de 10 centavos entregam os mesmos 30 centavos de troco!!!

Algoritmos Gulosos (Greedy): Troco em moedas

<u>IMPLEMENTAÇÃO E TEMPO DE EXECUÇÃO:</u>

- Para o algoritmo que escolhe uma moeda por vez e verifica os subproblemas, o tempo de execução é $\Theta(k)$, onde k é o número de moedas utilizadas na solução ótima.
- Como $k \leq n$, o tempo de execução é $O(n)$.

```
CASHIERS-ALGORITHM (x, c1, c2, ..., cn)

SORT n coin denominations so that 0 < c1 < c2 < ... < cn.
S ← ∅.   ← multiset of coins selected
WHILE (x > 0)
   k ← largest coin denomination ck such that ck ≤ x.
   IF (no such k)
      RETURN "no solution."
   ELSE
      x ← x − ck.
      S ← S ∪ { k }.
RETURN S.
```